When Every Token Counts: Optimal Segmentation for Low-Resource Language Models

Bharath Raj S [*], **Garvit Suri** [*], **Vikrant Dewangan** [*], **Raghav Sonavane** [†]

Sprinklr AI

Abstract

Traditional greedy tokenization methods have been a critical step in Natural Language Processing (NLP), influencing how text is converted into tokens and directly impacting model performance. While subword tokenizers like Byte-Pair Encoding (BPE) are widely used, questions remain about their optimality across model scales and languages. In this work, we demonstrate through extensive experiments that an optimal BPE configuration significantly reduces token count compared to greedy segmentation, yielding improvements in token-saving percentages and performance benefits, particularly for smaller models. We evaluate tokenization performance across various intrinsic and extrinsic tasks, including generation and classification. Our findings suggest that compression-optimized tokenization strategies could provide substantial advantages for multilingual and low-resource (LR) language applications, highlighting a promising direction for further research and inclusive NLP.

1 Introduction

The development of large language models (LLMs) has significantly advanced natural language processing. These models (Radford et al., 2019; Brown et al., 2020; OpenAI et al., 2024) have demonstrated unprecedented capabilities in tasks ranging from text generation and translation to complex problem-solving and creative writing. However, despite these advancements, challenges remain in effectively processing Low-Resource (LR) languages and optimizing models of varying scales.

A critical aspect influencing model performance is tokenization — the process of converting text into tokens that the model can understand. Tokenization methods are pivotal in large language models, with popular techniques including WordPiece (Schuster and Nakajima, 2012), SentencePiece (Kudo and Richardson, 2018), and Unigram-LM(Kudo, 2018). WordPiece, used in models like BERT (Devlin et al., 2019), tokenizes words into subword units based on their frequency in the training data, improving the model's handling of rare or out-of-vocabulary words. SentencePiece and Unigram-LM, commonly used in models like GPT, employ a character or byte-based approach that doesn't rely on predefined word boundaries, making them versatile across languages.

LR languages face two significant challenges in natural language processing: a lack of high-quality and diverse datasets and novel methods to represent this data. (Magueresse et al., 2020). Without ample data, models struggle to learn the complex linguistic patterns necessary for tasks such as machine translation, sentiment analysis, and summarization. Secondly, compression challenges in tokenization exacerbate the difficulties faced by LR languages. Common tokenization techniques, such as BPE, often fragment words into smaller, frequently occurring subwords. The bloating of tokens leads to higher computational and memory costs, as models must process longer sequences (Ahia et al., 2023). Inefficient tokenization also results in less accurate representations, leading to fragmented or improperly segmented tokens, which negatively impacts model performance in tasks requiring precise language understanding (Rust et al., 2021; Zhang et al., 2022). We refer to the strategy adopted by BPE as the Greedy segmentation algorithm.

The widely used GPT-2 tokenizer (Radford et al., 2019) handles any input without unknown tokens, yet it compromises tokenization efficiency, especially for non-English text and special characters. This English-centric model often splits languages like Turkish, Indonesian, or Malay into byte sequences, unnecessarily lengthening token sequences and reducing the effective context window for non-English content. While the GPT-4 (Ope-

[*]Equal Contribution

[†]corresponding author, raghavsonavane@gmail.com

Proceedings of the First Workshop on Language Models for Low-Resource Languages (LoResLM 2025), pages 320–334
January 20, 2025.

Language	`cl100k_base` Segmentation	English Translation	Tokenization Impact
English	p olic ym akers	policymakers	Breaks compound structure 'policy' (guidelines) + 'makers' (creators)
Turkish	y ü ks el me	rising/elevation	Base verb 'yük' (rise/load) splits into 'y ü k', loses connection to derivational 'sel' (become) and 'me' (action)
Malaysian	k ata c ak ları nd an	from what they will say	Future 'acak' (will) fragments into 'c ak', suffixes split into 'ları' (their) + 'dan' (from)
Finnish	ater i ak ok on ais u ude sta	from the material entirety	Compound splits: 'ateria' (meal) into 'ater i a', 'kokonaisuus' (entirety) into fragments, 'sta' (from) separates
Telugu Romanized	samb andh inchina	related to	Root 'samband' (relate) breaks into 'samb andh', separates from 'inchina' (past participle)
Tamil Romanized	kond iruk kire en	I am having/holding	Isolates 'iruk' (be), splits from 'kond' (having) and 'en' (I) markers
Hindi Romanized	pr ach in ak al	ancient times	Splits 'prachin' (ancient) into 'pr ach in', 'kaal' (time) becomes 'ak al'

Table 1: Segmentations produced by GPT-4's tokenizer `cl100k_base` across different language families, showing consistent patterns of morphological and phonological deterioration. Note that Romanized versions of Tamil, Telugu, and Hindi are shown to avoid byte interpretation.

nAI et al., 2024) tokenizer `cl100k_base` improves with a larger vocabulary and more diverse training data, it still shows biases in token distribution. For agglutinative languages (e.g., Turkish, Finnish) or languages with complex word structures, tokenization may create excessive token splits, impacting both efficiency and model performance. Examples of the inefficient segmentation of `cl100k_base` is shown in Table 1.

Motivated by the need to enhance tokenization strategies for LR languages and models of varying scales, we present an optimal BPE segmentation algorithm that reduces token counts, especially in morphologically complex and low-resource languages, achieving more efficient and meaningful segmentation. We demonstrate the algorithm's token-saving capacity across diverse languages, reducing token counts by 3-5% compared to greedy segmentation. This improvement is particularly impactful for rare and complex words, with compression rates increasing by up to 20%. Our comparative study reveals that models using optimal segmentation see up to a 10% increase in accuracy on downstream tasks, including text classification and generation.

2 Related Work

Recent research has focused on the effects that compression has on tokenization, which are particularly relevant for optimizing language models in resource-constrained environments. A study by (Goldman et al., 2024) shows the correlation that compression has on downstream tasks such as classification and generation. In contrast, (Uzan et al., 2024)'s exploration of greedy algorithms and (Schmidt et al., 2024)'s introduction of PathPiece have provided new insights into optimizing tokenization for both performance and efficiency, without looking into compression. Note that while (Uzan et al., 2024) and (Schmidt et al., 2024) demonstrate the effectiveness of their tokenizer, they show results on English tasks but do not show the impact on linguistic diversity. The paper by (Goldman et al., 2024) demonstrates this to some extent; however, their experiments focus primarily on English.

(Moghe et al., 2023) provide a task-oriented perspective on the challenges that LLMs encounter with low-resource languages, highlighting the need for tailored approaches in multilingual contexts. The quality of tokenization has been a subject of intense study, with comparative analyses by (Gallé, 2019), (Dagan et al., 2024), and (Saleva and Lignos, 2023) providing valuable insights into the relative performance of different tokenization methods across various languages and tasks. In multilingual settings, subword tokenizers lead to disproportionate fragmentation rates for different languages and writing script (Zhang et al., 2022). Similarly, monolingual optimized tokenizers may not be as efficient for multilingual settings (Rust et al., 2021). (Petrov et al., 2023) introduces a new concept known as parity or premiums in tokenizers which has shed light on the importance of balanced tokenization across

different languages in multilingual models. The disparities are particularly pronounced in African and Indian languages, as noted by (Myoya et al., 2023) and (Dongare, 2024; Velayuthan and Sarveswaran, 2024), respectively. While (Petrov et al., 2023) and (Velayuthan and Sarveswaran, 2024) demonstrate the critical role of tokenization in addressing challenges related to compression and parity in tokenization, they do not show the performance of LLMs on extrinsic tasks - especially for LR languages. These studies highlight the need for more inclusive tokenization and pre-training strategies that can serve diverse linguistic communities.

Our work shows that by improving tokenization methods - specifically compression - we can achieve performance on extrinsic tasks on LR languages. Our approach allows us to optimize inference time and cost and have an equally good as the original tokenization. (Ahia et al., 2023) also highlights the economic implications of these disparities, comparing the pricing of language model usage across different languages and revealing systemic biases in current NLP technologies.

3 Background

We first provide a brief description of the steps involved in tokenization that is pre-tokenization, vocabulary construction, and segmentation. We then describe the Token Saving Ratio (TSR) metric used to compare results throughout our paper.

3.1 Stages of Tokenization

In any modern natural language system, a document d, before it gets encoded into a set of tokens $\{t_1, t_2, \ldots t_K\}$ goes through 3 main stages to tokenization. They are (i) Pre-tokenization (ii) Vocabulary Construction and (iii) Segmentation. Pre-tokenization consists of the initial processing phase where raw text in the document undergoes fundamental transformations. It ensures the text is in a consistent format for subsequent processing. The vocabulary construction phase focuses on building a comprehensive token dictionary V of size m from the processed text. This stage involves analyzing large text corpora to identify recurring patterns and meaningful units. The system conducts frequency analysis to determine the most common patterns and handles rare words appropriately. The final segmentation stage implements the actual tokenization process using the constructed vocabulary.

Given a vocabulary V, and a document d, segmentation task S refers to the task of dividing the document d into a sequence of tokens (t_i), such that $S(d) = \{t_1, \ldots t_K | \forall i \in [1, K], t_i \in V\}$. During this phase, the system applies specific tokenization rules to convert text into its final token form. The process includes mechanisms for handling unknown tokens (UNK) that may not exist in the vocabulary. Subword tokenization strategies are implemented to manage complex words and maintain semantic meaning. The stage concludes with the assignment of unique token IDs to each segmented unit, creating the final tokenized representation of the text. This standardized format enables efficient processing in downstream natural language processing tasks. For the scope of this work, we exclusively study the segmentation stage of tokenization and detail an optimal segmentation algorithm.

3.2 Token Saving Ratio (TSR)

To measure the quality of segmentation, we define and use the metric, Token Saving Ratio (TSR), to capture the ratio of tokens saved when using tokenizer T_A with segmentation strategy S_A compared to tokenizer T_B with strategy S_B. The Token Saving Ratio when using tokenizer T_A compared to tokenizer T_B is defined as:

$$TSR = \frac{|S_B(d)| - |S_A(d)|}{|S_B(d)|} \quad (1)$$

A positive TSR directly translates to shorter sequence lengths, which is paramount for computational efficiency. Since the computational complexity of transformer-based models typically scales quadratically with sequence length ($O(n^2)$), reducing the number of tokens can significantly decrease both memory requirements and processing time. For instance, if tokenizer T_A produces sequences half the length of T_B, the computational cost could potentially be reduced by a factor of four.

4 Optimal Segmentation

In this section, we define the problem of optimal segmentation mathematically and follow it up with a discussion of our algorithm presented in Algorithm 1.

4.1 Definition

Given a vocabulary V of size m, we define optimal segmentation (S^*) as the segmentation that minimizes the number of tokens a given document d can be split into. Formally,

$$S(d) = \{t_1, \ldots t_K | t_i \in V\}$$
$$S^* = \underset{S}{\text{minimize}} \, |S(d)| \quad (2)$$

4.2 The Algorithm

We use a dynamic programming formulation similar to the Viterbi algorithm (Forney, 1973) and produces the optimal segmentation S^*. Given a document d, define $dp[i]$ as the minimal number of tokens needed to segment the prefix $d_0 d_1 \ldots d_i$ (positions 0 to i, inclusive). We set $dp[-1] = 0$ as the base case, representing the empty prefix requiring zero tokens. The parent array par serves as a backtracking mechanism where $par[i]$ points to the end of the previous token in the optimal segmentation.

Algorithm 1: Algorithm for finding optimal segmentation S^*

Input:
$B = [B_0, B_1, \ldots, B_{n-1}] \in \Sigma^*$ {byte sequence}
$V \subset \Sigma^*$, {vocabulary}
$\mathcal{T}(V^R)$ with root r {trie on reversed vocabulary V^R}
Define:
$\delta(v) : \mathcal{T} \to \mathcal{T} \cup \{\emptyset\}$ {outputs child of v in trie $\mathcal{T}$}
I : $V \to \{True, False\}$ {indicator function detecting if node is terminal node}
Output: $S^* \in V^*$ {optimal segmentation}
Initialize:
$dp[i] \leftarrow i + 1, \forall i \in [0, n-1]; dp[n] \leftarrow 0$
$par[i] \leftarrow i - 1, \forall i \in [0, n-1]$ {parent array}
for $i \in [0, n-1]$ **do**
  $v \leftarrow r$
  for $j = i \downarrow 0$ **do**
    $v \leftarrow \delta(v, B[j])$ {child of node v corresponding to $B[j]$}
    if $v = \emptyset$ **then**
      break
    end if
    if I$(v) \wedge (dp[j-1] + 1 < dp[I])$ **then**
      $dp[i] \leftarrow dp[j-1] + 1$
      $par[i] \leftarrow j - 1$
    end if
  end for
end for
$S \leftarrow \emptyset$ {initialize empty sequence}
$k \leftarrow n - 1$
while $k \neq -1$ **do**
  $S \leftarrow S \cup \{B[par[k] + 1 : k + 1]\}$ {$B[i:j]$ denotes substring}
  $k \leftarrow par[k]$
end while
return S^R {reversed sequence}

The recurrence relation is:

$$dp[i] = \min_{(0 \le j \le i)} (dp[j-1] + 1) \quad (3)$$
$$\text{where } d_j d_{j+1} \ldots d_i \in V$$

It should be noted that multiple values of j can lead to the optimal value for $dp[i]$. In such cases, Algorithm 1 only considers the largest such j i.e., only the smallest suffix is considered. Once the dynamic programming array dp is calculated, we use the state transitions to find the optimal segmentation (S^*). A detailed proof of the optimality of this algorithm can be found in Appendix B. Additionally, to efficiently check the condition $d_j d_{j+1} \ldots d_i \in V$, we use a Trie data structure built on the reversed tokens of the vocabulary V and is denoted by its root node $root$ in the algorithm.

Given that the length of the longest token in the vocabulary V is M, and the length of the document d is N, the worst-case time complexity of our algorithm is $O(NM)$ which is the same as the worst case time complexity of the greedy segmentation used in the commonly available BPE tokenizer implementations. The greedy segmentation algorithm stores the vocabulary V and the merges made during vocabulary creation, leading to a space complexity of $O(\sum |t_i|) | t_i \in V$. In our algorithm, we store the vocabulary V and the Trie data structures built on the reversed tokens of V, leading to the same space complexity.

Through our extensive experimentation described in the next sections, we showcase the effectiveness of our algorithm in improving the TSR. We also show improvements in downstream performance.

5 Experimental Setup

For our work, we extended on OpenAI's[1] family of Tokenizers which are available in three distinct vocabulary sizes: 50K, 100K, and 200K tokens, as detailed in Table 3. In this study, we rely on the original pre-tokenization regular expressions and the trained vocabulary made public by OpenAI, without making any modifications to it. Our study concentrated exclusively on the segmentation strategies of these tokenizers.

We divide our experiments into two parts: **intrinsic** and **extrinsic**, following the approach of (Goldman et al., 2024). The intrinsic experiments focus purely on the segmentation aspect of tokenization, without involving any deep learning models. Here, we analyze the TSR when comparing optimal versus greedy segmentations across languages. Based on vocabulary size, we select appropriate tokenizers according to Table 3, which serve as the baseline

[1] https://github.com/openai/tiktoken/blob/main/tiktoken_ext/openai_public.py

Language	Greedy	Optimal	TSR (%)	Tokenization Impact
English	p olic ym akers	policy makers	50	Respects natural compound boundary of 'policy' (guidelines) + 'makers' (creators) vs. meaningless 'p olic'
	sk ys canner	sky scanner	33	Preserves 'sky' (aerial) + 'scanner' (reader) vs. invalid 'sk ys' split
Indonesian	mung kink ah	mungkin kah	33	Separates 'mungkin' (possible) and 'kah' (question marker) vs. invalid 'kink'
Turkish	y ü ks el me	yük sel me	40	Maintains 'yük' (rise) + 'sel' (become) + 'me' (action) vs. broken 'y ü ks'
Malaysian	k ata c ak ları nd an	kat acak ların dan	43	Preserves 'acak' (future) + 'ların' (their) + 'dan' (from) vs. 'c ak ları nd'
Finnish	f otos y nt ees ille	foto syn tees ille	33	Keeps 'foto' (light) + 'syn' (with) + 'ille' (for) vs. broken 'f otos y nt'
	dat apro j ek tor	data proj ekt ori	33	Retains 'data' (data) + 'projekt' (project) vs. invalid 'j ek tor'
Telugu	Sang arsh ana	Sangars hana	33	Maintains 'Sangarsh' (struggle) + 'ana' (action) vs. 'Sang arsh'
	Mall igad u	Malliga du	33	Separates 'Malliga' (name) + 'du' (masculine) vs. 'igad'
Tamil	puri yav illai	puriya villai	33	Preserves 'puriya' (understand) + 'villai' (not) vs. 'yav'
	yend rav udan	yendra vudan	33	Maintains 'yendra' (saying) + 'vudan' (with) vs. 'rav'
Hindi	v ich ar sh il	vi chars hil	40	Keeps 'vichar' (thought) + 'shil' (having quality) vs. 'v ich ar'
	pr ach in ak al	pra china kal	40	Retains 'prachin' (ancient) + 'kal' (time) vs. 'pr ach in'

Table 2: Comparison of BPE segmentation modes showing linguistically motivated vs. arbitrary tokenization breaks. TSR (Token Stability Ratio) indicates the percentage improvement in segmentation quality.

T_B in Equation 1 for evaluating the TSR. For the extrinsic experiments, we investigate how TSR affects decoder-only models, specifically examining its impact on the perplexity and accuracy of the models listed in Section 5.3 across various tasks.

Tokenizer	Vocab Size (m)
gpt-2	50K
cl100k_base	100K
o200k_base	200K

Table 3: OpenAI Tokenizers and Their Configurations

5.1 Intrinsic Evaluation Datasets

For performing the intrinsic evaluation, we used the CC-100 dataset (Wenzek et al., 2020). The CC-100 dataset consists of monolingual data of 116 languages extracted from the January-December 2018 Commoncrawl snapshots. We benchmark on the English language using the Wikipedia corpus readily accessible on Kaggle Datasets[2]. We utilized the Wikipedia 2023 dump, which contains 6 million articles, titles, text, and categories.

[2] https://www.kaggle.com/datasets/jjinho/wikipedia-20230701

5.2 Extrinsic Evaluation Tasks

We relied on the intrinsic evaluation of languages to choose the languages for our extrinsic experiments. We choose English to show that there is no degradation in performance in a language with near-zero compression. We also chose Finnish, Indonesian, and Turkish which show up in the top languages with high TSR. To evaluate our pre-trained checkpoints, we evaluated multiple tasks for different languages, as detailed in Table 4. The tasks are mentioned in detail one by one below in Appendix D. For all of the extrinsic experiments, we set the vocabulary size to $m = 50K$ and use the gpt-2 tokenizer (Table 3).

To highlight the impact of TSR, we also repeat the evaluation on a subset of each dataset where there is a non-zero TSR. We denote this subset by TSR^*. We split each dataset into two groups: the full dataset (All) and a subset containing only examples where Greedy and Optimal segmentation produce different token sequences (TSR^*). This division allows us to isolate and better understand the impact of segmentation strategies on samples where the tokenizer makes different decisions. The split is highlighted in the Table 5 where we denote the percentage of samples used to construct the

Language	Task Name	Task Type
English	Penn-Tree Bank (Marcus et al., 1993)	Generation
English	LAMBADA (Paperno et al., 2016)	Generation
English	QQP [3]	Classification
English	Story Cloze (Mostafazadeh et al., 2016)	Classification
Finnish	TyDiQA-GoldP (Clark et al., 2020)	Classification
Indonesian	Emot (Saputri et al., 2018)	Classification
Indonesian	WreTe (Setya and Mahendra, 2018)	Classification
Turkish	XNLI (Conneau et al., 2018)	Classification

Table 4: Tasks for Different Languages

TSR^* dataset.

Language	Dataset Name	Non-zero TSR
English	QQP	4.69
English	Story Cloze	6.15
Finnish	TyDiQA-GoldP	**62.20**
Indonesian	Emot	**88.64**
Indonesian	WreTe	**75.00**
Turkish	XNLI	**100.00**

Table 5: Percentage of samples with non-zero TSR across datasets, used to create the TSR^* split.

5.3 Baselines

For our extrinsic evaluations we use two sizes of the GPT-2 (Radford et al., 2019) language models, comprising 120 million and 350 million parameters, fine-tuned on the extrinsic fine-tuning dataset. We fine-tune the 120M and 350M versions of the GPT-2 model on the OpenWebText dataset and use it for all the downstream tasks. We did not do a complete pretraining from scratch as the model pre-trained with greedy segmentation only has to learn the difference in the distribution of tokens with optimal segmentation. Detailed model configurations and hyper-parameters are provided in Appendix A.

6 Results

In this section, we present the results of intrinsic evaluation on the CC-100 dataset. We first highlight qualitative examples to showcase the inefficiency of BPE with Greedy segmentation compared to BPE with Optimal segmentation We also showcase an interesting observation that word length has on the TSR. Finally, to validate our optimal segmentation algorithm, we conduct extensive extrinsic evaluations across multiple downstream tasks. First, we report improvement upon Greedy BPE's performance across language boundaries for non-English tasks. At the same time, we report an increase in improvements for the TSR^* split of the dataset, thus highlighting the need for token saving in downstream performance. At the end, we report perplexity scores on English datasets to state that the improvement provided by our optimal segmentation doesn't reduce the tokenizer's performance in English.

6.1 Intrinsic Evaluation

6.1.1 Qualitative Results

Table 2 presents examples of how different tokenizers segment the same vocabulary in distinct ways, depending on their inference mode. Greedy BPE for instance, splits the word "policy makers" into 4 tokens: "p" "olic" "ym" "akers", while the optimal segmentation splits it into two tokens: "policy" and "makers". The table illustrates fundamental linguistic issues with greedy BPE segmentation across different language families. In English, it fails to respect compound word boundaries (*policymakers*). For agglutinative languages like Turkish and Malaysian, it breaks crucial morphological units, splitting tense markers and case endings arbitrarily. In Dravidian languages (Telugu, Tamil), it fails to preserve verb roots and aspectual markers. For Indo-Aryan languages, it incorrectly segments Sanskrit-derived compounds, creating linguistically meaningless units. These issues extend beyond mere segmentation - they affect the model's ability to learn proper morphological patterns, potentially impacting downstream task performance. While BPE has been widely adopted for its computational efficiency, these examples demonstrate the need for more linguistically-informed tokenization strategies that respect language-specific morphological structures that our optimal segmentation can provide.

6.1.2 Quantitative Results

We report TSR across the 116 languages in the CC-100 dataset. Languages with the highest TSR can be found in Table 6. This table demonstrates the wide variation in TSR achieved by tokenizing different languages across 50K, 100K, and 200K vocabulary sizes. The languages with the highest TSR, such as Oromo, Swati, and Quechua, maintain over 4.5% TSR even at the largest 200K vocabulary. In contrast, lower-resourced languages like Tagalog, Bosnian, Hausa, and Turkish have lower compression rates, near 3% even at the smaller 50K

Language	TSR (in %)		
	50K	100K	200K
Quechua	4.74	5.09	4.81
Oromo	4.72	5.27	3.02
Basque	4.53	4.06	3.56
Zulu	4.49	4.74	3.61
Xhosa	4.24	4.65	3.46
Swati	4.14	5.17	3.63
Telugu Romanized	4.13	3.76	3.75
Malay	4.05	2.73	1.72
Tamil Romanized	3.99	4.22	4.20
Indonesian	3.83	2.43	1.58
Finnish	3.80	4.32	3.37
Swahili	3.74	3.73	2.37
Somali	3.59	4.51	2.59
Malagasy	3.57	3.54	2.38
Uzbek	3.57	4.30	3.52
Hausa	3.52	3.83	1.51
Estonian	3.45	4.10	3.18
Bosnian	3.40	2.65	2.08
Tagalog	3.38	2.56	1.47
Turkish	2.90	2.88	2.62

Table 6: TSR on LR Languages: 20 languages with highest TSR for different vocabulary sizes (m) as 50K, 100K, and 200K.

size. This data offers important insights to guide vocabulary selection and optimization decisions, particularly for deploying efficient language models in resource-constrained environments targeting LR languages.

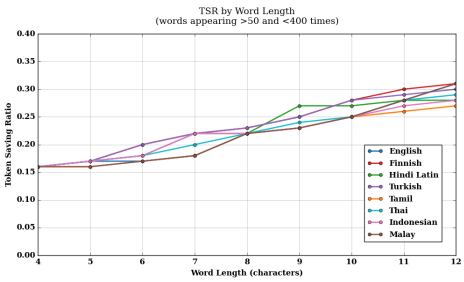

Figure 1: TSR and Word length correlation across seven different languages, with Vocab. size $m = 100K$.

Word length Relation with TSR: We plot an interesting observation that word length has with TSR in Figure 1. We notice a strong correlation, with longer words achieving better compression ratios (increasing from ∼0.15 for 4-character words to ∼0.30 for 11-12 character words) - suggesting that word length appears to be one of the factors in compression efficiency across these linguistically diverse languages. This pattern is consistent across all languages in our study, though with varying slopes - Finnish and Turkish show steeper increases with word length, while English demonstrates a more gradual rise. Notably, agglutinative languages like Finnish, Turkish, and Indonesian, which typically have longer words due to their morphological structure, benefit more from our optimal segmentation strategy as word length increases. Thai shows a moderate but steady increase despite its analytic nature and lack of explicit word boundaries, and Tamil, with its complex agglutinative morphology, displays a more gradual rise similar to English, possibly due to its unique script-to-byte conversion patterns.

6.2 Extrinsic Evaluation Tasks

Table 7 presents a systematic analysis across languages and tasks, examining how different types of tokenization errors—particularly compound word splitting, verb root identification, and morpheme boundary detection—affect downstream performance. For the Indonesian Emot task with the 120M model, optimal segmentation improves accuracy by 4.32% (from 40.23% to 44.55%) in the full dataset. This improvement becomes more pronounced in the TSR^* subset, reaching **5.64%** (from 39.23% to 44.87%), primarily due to better handling of compound words (e.g., "memberikan" → "memberi" + "kan") and proper verb root preservation. In the 350M model, while the overall gap is smaller at 2.50%, it still increases to **2.56%** in the TSR^* subset, showing similar error patterns but at reduced magnitudes. The WreTe task shows similar error patterns: optimal segmentation yields a 2.00% improvement in the full dataset, expanding to **2.66%** in TSR^*, with compound word splitting errors driving a significant portion of the performance difference. For Turkish (XNLI), we observe improvements of 0.56% to **0.76%** (120M) and 0.98% to **0.79%** (350M), where analysis shows that agglutinative morpheme boundaries (particularly case markers and possessive suffixes) significantly impact performance. Finnish presents a unique case where accuracies remain identical between *All* and TSR^* subsets, as all words exhibit non-zero TSR scores.

For English tasks, we observe moderate differences in the performance between *All* and the TSR^* subset. In Story Cloze, the 350M model shows an improvement with Optimal segmentation in TSR^* (**7.83**% gain, from 52.17% to 60.00%) compared to the full dataset (**0.43%** gain, from 51.31% to 51.74%). QQP shows varying patterns: in the 120M model, Greedy performs better in both sets, with the gap being more pronounced in TSR^* (-

Size	Method	English				Finnish		Indonesian				Turkish	
		QQP		Story Cloze		TyDiQA-GoldP		Emot		WreTe		XNLI	
		All	TSR*	All	TSR*	All	TSR*	All	TSR*	All	TSR*	All	TSR*
120M	Greedy	**75.22**	**81.58**	**51.90**	**57.39**	82.91	82.91	40.23	39.23	76.00	70.67	64.35	63.83
	Optimal	74.52	81.20	51.31	52.17	**83.76**	**83.76**	**44.55**	**44.87**	**78.00**	**73.33**	**64.91**	**64.59**
350M	Greedy	**76.34**	**83.46**	51.31	52.17	85.47	85.47	43.18	41.54	**78.00**	74.67	65.35	65.27
	Optimal	74.73	81.83	**51.74**	**60.00**	**85.90**	**85.90**	**45.68**	**44.10**	**78.00**	**76.00**	**66.33**	**66.06**

Table 7: GPT-2 Accuracy Results on Multiple Datasets. TSR* columns show results on the non-zero TSR subset.

0.70% vs -0.38%). These results suggest that evaluating the TSR^* subset often amplifies the impact of the segmentation strategy, particularly for tasks where token sequencing plays a crucial role. The better performance of Greedy might be attributed to English's relatively straightforward morphological structure compared to agglutinative languages like Turkish or Finnish. English words typically have clearer boundaries and less complex internal structure, allowing the tokenization strategies to focus on semantic units rather than navigating complex morphological combinations.

Model Size	Segmentation	Perplexity (↓)
120M	Greedy	43.76
	Optimal	**39.97**
350M	Greedy	34.56
	Optimal	**34.45**

Table 8: GPT-2 Perplexity on English datasets (lower is better)

We also report the perplexity metric evaluation on English datasets (LAMBADA) to show that our Optimal segmentation does not substantially degrade model performance compared to Greedy segmentation. We present this result in the Table 8. We report that the differences in perplexity are minimal. These results suggest that our proposed tokenization strategy maintains comparable modeling capability on English text, indicating that the improvements we observe on non-English tasks are not achieved at the expense of English language modeling quality.

7 Conclusion

In the scope of this work, we identified the inefficient greedy segmentation method used in the BPE tokenizer and proposed an optimal segmentation algorithm that results in efficient token utilization, particularly for LR languages. We established the optimality of our algorithm by showing its impact in both intrinsic and extrinsic experiments as done in the literature. By studying multiple languages, we observed a strong correlation between improvements in Token Saving Ratios and linguistically better segments, with this effect being especially pronounced for morphologically complex words and propagating to performance improvement in downstream tasks. These findings underscore the need for research in tokenization approaches that can boost model effectiveness, especially for language models serving low-resource languages.

8 Limitations and Future Work

Our work demonstrates the impact of using BPE tokenization with optimized segmentation on tokenization efficiency across multiple languages. Although we evaluated models on intrinsic metrics for a variety of languages, our extrinsic evaluations focused primarily on four languages: English, Finnish, Indonesian, and Turkish. We chose these languages to capture diversity in typology and morphology, as well as to leverage the relatively richer resources available for them compared to many other LR languages. In the future, we intend to perform a more comprehensive follow-up study to replicate these findings across a wider array of languages provided in Table 6, aiming to validate the broader applicability of our approach. This could help assess the robustness of using optimal segmentation across languages with more complex or less studied morphological characteristics.

Future research would also explore other underlying factors influencing tokenization quality and its broader impact on language model success. This extension would help us understand whether our findings about optimal segmentation scale to models with larger vocabularies and more sophisticated architectures. In future work, we plan to extend our analysis to larger foundation models like LLaMA-3 (Grattafiori et al., 2024), where the impact of tokenization strategies may reveal additional insights about segmentation in more complex architectures. We would also explore improvements in

other stages, such as optimal vocabulary selection and encoding methods for adaptive tokenization.

References

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David R. Mortensen, Noah A. Smith, and Yulia Tsvetkov. 2023. Do all languages cost the same? tokenization in the era of commercial language models. *Preprint*, arXiv:2305.13707.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Gautier Dagan, Gabriele Synnaeve, and Baptiste Rozière. 2024. Getting the most out of your tokenizer for pre-training and domain adaptation. *ArXiv*, abs/2402.01035.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Pratibha Dongare. 2024. Creating corpus of low resource Indian languages for natural language processing: Challenges and opportunities. In *Proceedings of the 7th Workshop on Indian Language Data: Resources and Evaluation*, pages 54–58, Torino, Italia. ELRA and ICCL.

G.D. Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Matthias Gallé. 2019. Investigating the effectiveness of BPE: The power of shorter sequences. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1375–1381, Hong Kong, China. Association for Computational Linguistics.

Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. 2024. Unpacking tokenization: Evaluating text compression and its correlation with model performance. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 2274–2286, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj

Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *ArXiv*, abs/1804.10959.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Conference on Empirical Methods in Natural Language Processing*.

Alexandre Magueresse, Vincent Carles, and Evan Heetderks. 2020. Low-resource languages: A review of past work and future challenges. *Preprint*, arXiv:2006.07264.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, 19(2):313–330.

Nikita Moghe, Evgeniia Razumovskaia, Liane Guillou, Ivan Vulić, Anna Korhonen, and Alexandra Birch. 2023. Multi3NLU++: A multilingual, multi-intent, multi-domain dataset for natural language understanding in task-oriented dialogue. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3732–3755, Toronto, Canada. Association for Computational Linguistics.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *Preprint*, arXiv:1604.01696.

Rozina Myoya, Fiskani Banda, Vukosi Marivate, and Abiodun Modupe. 2023. Fine-tuning multilingual pretrained african language models. In *AfricaNLP*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word pre-

diction requiring a broad discourse context. *Preprint*, arXiv:1606.06031.

Aleksandar Petrov, Emanuele La Malfa, Philip H. S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. *Preprint*, arXiv:2305.15425.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *Comput. Linguist.*

Sebastian Ruder, Noah Constant, Jan Botha, Aditya Siddhant, Orhan Firat, Jinlan Fu, Pengfei Liu, Junjie Hu, Dan Garrette, Graham Neubig, and Melvin Johnson. 2021. XTREME-R: Towards more challenging and nuanced multilingual evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10215–10245, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

Jonne Saleva and Constantine Lignos. 2023. What changes when you randomly choose BPE merge operations? not much. In *Proceedings of the Fourth Workshop on Insights from Negative Results in NLP*, pages 59–66, Dubrovnik, Croatia. Association for Computational Linguistics.

Mei Silviana Saputri, Rahmad Mahendra, and Mirna Adriani. 2018. Emotion classification on indonesian twitter dataset. In *Proceedings of the 2018 International Conference on Asian Language Processing(IALP)*, pages 90–95. IEEE.

Craig W. Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. 2024. Tokenization is more than compression. *ArXiv*, abs/2402.18376.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Ken Nabila Setya and Rahmad Mahendra. 2018. Semi-supervised textual entailment on indonesian wikipedia data. In *Proceedings of the 2018 International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*.

Omri Uzan, Craig W. Schmidt, Chris Tanner, and Yuval Pinter. 2024. Greed is all you need: An evaluation of tokenizer inference methods. *ArXiv*, abs/2403.01289.

Menan Velayuthan and Kengatharaiyer Sarveswaran. 2024. Egalitarian language representation in language models: It all begins with tokenizers. *Preprint*, arXiv:2409.11501.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Bryan Wilie, Karissa Vincentio, Genta Indra Winata, Samuel Cahyawijaya, Xiaohong Li, Zhi Yuan Lim, Sidik Soleman, Rahmad Mahendra, Pascale Fung, Syafri Bahar, and Ayu Purwarianti. 2020. IndoNLU: Benchmark and resources for evaluating Indonesian natural language understanding. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 843–857, Suzhou, China. Association for Computational Linguistics.

Shiyue Zhang, Vishrav Chaudhary, Naman Goyal, James Cross, Guillaume Wenzek, Mohit Bansal, and Francisco Guzman. 2022. How robust is neural machine translation to language imbalance in multilingual tokenizer training? In *Proceedings of the 15th biennial conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 97–116, Orlando, USA. Association for Machine Translation in the Americas.

A Language Model Parameters

The $120M$ parameter models were trained using the GPT architecture with the following parameters.

Model	Dim.	Heads	Layers	Batch	Seq Len
120M	1024	16	24	1024	1024
350M	2048	8	16	2048	1024

Table 9: Model Configurations

B Proof of Optimality

B.1 Dynamic Programming Formulation

Define $dp[i]$ as the minimal number of tokens needed to segment the prefix $S_0S_1 \dots S_i$ (positions 0 to i, inclusive). We set $dp[-1] = 0$ as the base case, representing the empty string requiring zero tokens. The recurrence relation is:

$$dp[i] = \min_{(0 \le j \le i)} (dp[j-1] + 1)$$
$$\text{where } S_jS_{j+1} \dots S_i \in V$$

B.2 Proof by Contradiction:

Suppose there exists a segmentation of the prefix $S_0S_1 \dots S_i$ into tokens from vocabulary V that uses fewer tokens than $dp[i]$ computed by our algorithm.

Let this supposed optimal segmentation divide the prefix into tokens, ending at positions $-1 = k_{-1} < k_0 < k_1 < k_2 < \dots < k_{m-1} = i$, resulting in m tokens:

$$\begin{aligned} T_0 &= S_{k_{-1}+1}S_{k_{-1}+2} \dots S_{k_0}, \\ T_1 &= S_{k_0+1}S_{k_0+2} \dots S_{k_2}, \\ &\vdots \\ T_{m-1} &= S_{k_{m-2}+1}S_{k_{m-2}+2} \dots S_{k_{m-1}}. \end{aligned}$$

Each $T_j \in V$, and the total number of tokens is $m < dp[i]$.

Consider the last token T_{m-1} in this segmentation, which covers the substring $S_{k_{m-2}+1}S_{k_{m-2}+2} \dots S_{k_{m-1}}$. Since $T_{m-1} \in V$, our algorithm, when computing $dp[i]$, examines this possibility.

By the definition of our algorithm:

$$dp[i] = \min\left(dp[i],\ dp[k_{m-2}] + 1\right)$$

In the worst case, there are no better alternatives than k_{m-2},

$$dp[i] = dp[k_{m-2}] + 1$$

By a similar argument,

$$\begin{aligned} dp[k_{m-2}] &= dp[k_{m-3}] + 1, \\ dp[k_{m-3}] &= dp[k_{m-4}] + 1, \\ &\vdots \\ dp[k_i] &= dp[k_{i-1}] + 1, \\ &\vdots \\ dp[k_0] &= dp[k_{-1}] + 1, \end{aligned}$$

Using the above results,

$$\begin{aligned} dp[i] &= dp[k_{m-2}] + 1 \\ &= dp[k_{m-3}] + 1 + 1 \\ &\vdots \\ &= dp[k_i] + m - 1 - i \\ &\vdots \\ &= dp[k_{-1}] + m - 1 - (-1) \\ &= m \end{aligned}$$

Simplifying to,

$$dp[i] = m$$

However, we initially assumed that $m < dp[i]$. This leads to a contradiction, which means our initial assumption that there exists a better segmentation is wrong. This completes the proof.

C Intrinsic Statistical Analysis

Frequency analysis with Word length: The word frequency distribution pattern provides crucial context for interpreting the extrinsic task performance. The frequency-based analysis shown in Fig. 2 helps explain why the impact of optimal segmentation varies significantly across languages and tasks, with larger gains in languages where optimal segmentation of longer words, though less frequent, carries greater semantic importance. The reported token saving percentages (TSR) may underestimate the true potential of optimal segmentation due to frequency-based evaluation bias. Since longer words (>6 characters) occur substantially

less frequently in the corpus, their improvements in segmentation quality are numerically diluted in aggregate metrics. Many of these longer words often carry crucial semantic information through compound formation and morphological processes, as evidenced in Table 2.

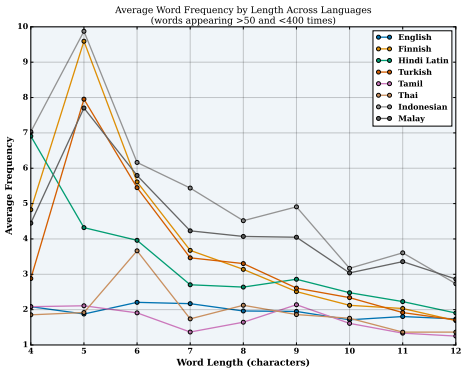

Figure 2: Frequency vs Word Length: Comparison across seven languages with a vocab size of $m = 100K$

In-context evaluation: Figure 3 compares the token-saving performance of greedy and optimal segmentations across different languages as the number of in-context examples increases. It shows significant variation in the token saving percentages between languages, with the Optimal tokenizer outperforming the Greedy approach. The gap between the two tends to widen as more examples are provided, indicating a better ability from a language model to leverage contextual information. This visualization offers valuable insights into the intrinsic multilingual capabilities of these tokenizers, which can inform decisions around model architecture and deployment for multilingual applications.

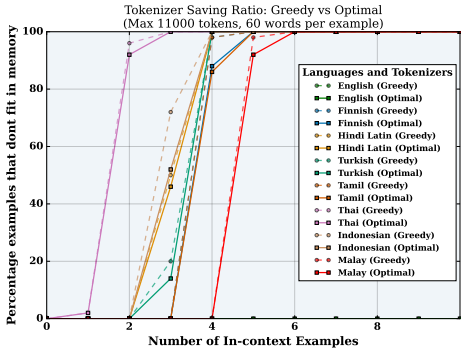

Figure 3: In-Context Comparison: Percentage of examples that fit across languages with vocab size of $m = 100K$, highlighting the impact on extrinsic performance with increasing in-context examples.

D Extrinsic Evaluation Tasks

We describe the different tasks used for fine-tuning our models:

- For English generation tasks, we used the Penn Tree Bank (PTB) dataset (Marcus et al., 1993), which serves as a traditional benchmark for assessing language generation capabilities through zero-shot perplexity, leveraging its pre-internet content. Additionally, the LAMBADA dataset (Paperno et al., 2016) was employed to test the model's ability to comprehend and predict the last word in a paragraph, challenging its handling of long-range dependencies. For English classification tasks, we utilized the Quora Question Pairs (QQP) dataset [4]), which involves determining if question pairs are duplicates, evaluated using the F1 metric. The Story Cloze dataset (Mostafazadeh et al., 2016) was also used to measure the model's ability to choose the correct ending for short narratives, further assessing classification performance.

- For Finnish we used the gold passage version of the Typologically Diverse Question Answering dataset (**TyDiQA-GoldP**) (Clark et al., 2020) (Ruder et al., 2021). It consists of a question, a relevant passage, and an answer - yes or no.

- Expanding to Indonesian, we employed two datasets from the indoNLU (Wilie et al., 2020; Saputri et al., 2018; Setya and Mahendra, 2018) collection : EmoT, which is an emotion classification dataset collected from Twitter consisting of tweets in Indonesian covering five emotion labels: anger, fear, happiness, love, and sadness; and WReTE, which is a textual entailment dataset constructed from Wikipedia revision history, containing pairs of sentences with binary semantic relations .

- For Turkish, the XNLI dataset (Conneau et al., 2018) was utilized. XNLI extends the MultiNLI dataset into a multilingual evaluation suite, providing a benchmark for cross-lingual language understanding through

[4] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

sentence-pair classification tasks across 15 languages.